

# How to customize LTSI Test Project with LTP

Kenji TADANO and Kengo IBE  
Mitsubishi Electric  
13th, November 2015  
Japan Technical Jamboree 55

# Outline

---

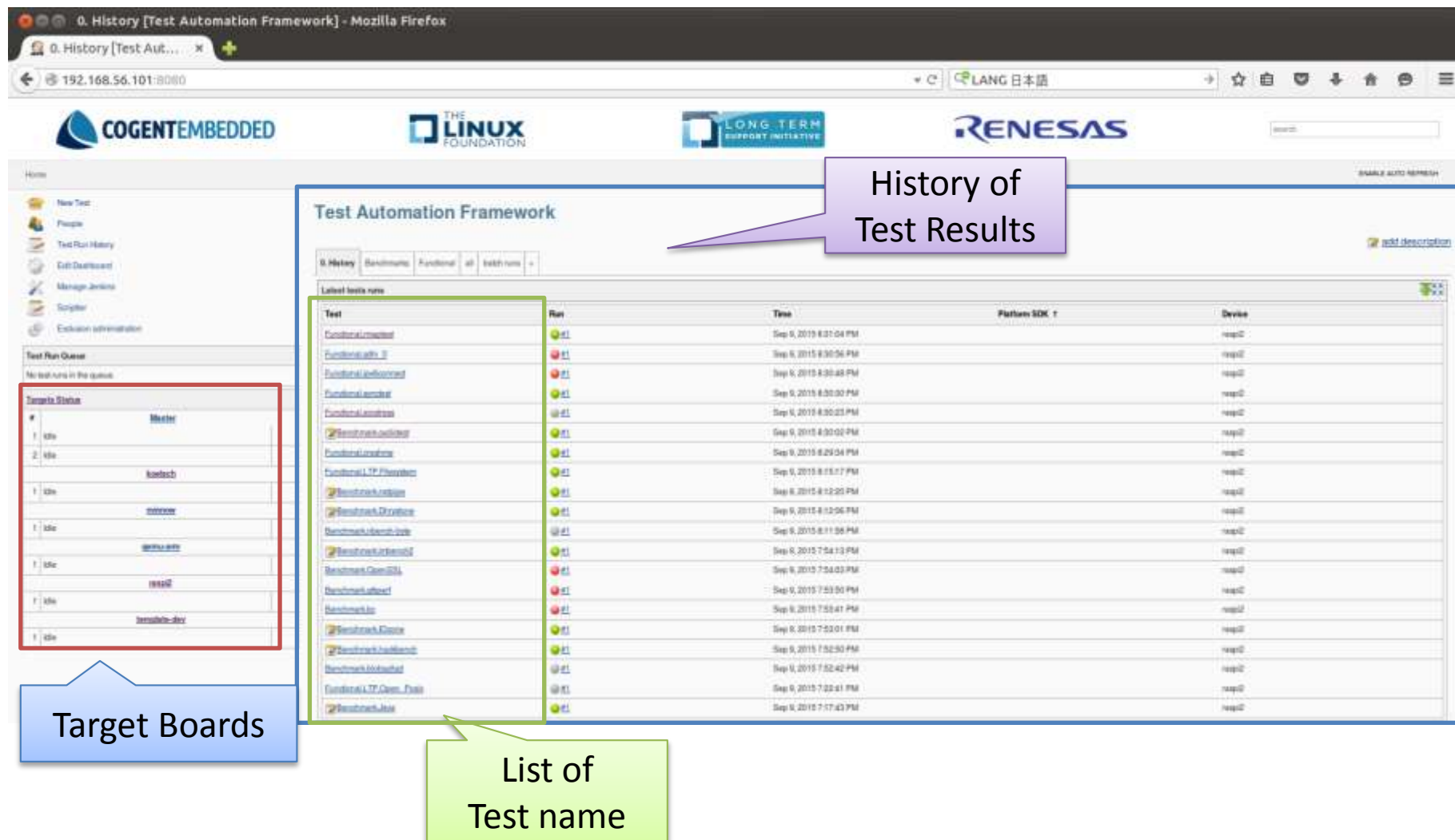
- What is the LTSI Project?
  - LTSI Test Environment
- How to Customize ?
  - Add New Board (e.g. Raspberry Pi2)
  - Add New Test Suite (e.g. LTP : Linux Test Project)
- Comparison result of running LTP for each case
- Conclusion

# What is LTSI Test Project?

- LTSI Project :
  - The project creates and maintains Linux Kernel which is expected to be stable in quality for the typical lifetime of a consumer electronics product, typically 2 years.
  - LTSI-4.1 Developing now
    - **Closed Merge Window:** End of October
- LTSI Test Project
  - The project creates the LTSI Test Environment .
  - The LTSI Test Environment is Jenkins based automation test framework.
  - Include many test suites and kinds of target boards
    - 28 benchmarks and 33 functional test programs are already integrated
    - Minnow board(x86), koelsch(arm), quem-arm(QEMU) are already integrated
  - I hope to further increase the kind of target board, test suite.
  - I'm happy that many people will join this project.

# LTSI Test Environment(Overview)

- The main page of GUI of LTSI Test Environment



0. History [Test Automation Framework] - Mozilla Firefox

0. History [Test Aut...]

192.168.56.101:8080

LANG 日本語

COGENT EMBEDDED THE LINUX FOUNDATION LONG TERM SUPPORT INITIATIVE RENESAS

Home

New Test People Test Run History Test Dashboard Manage Jenkins Scripter Execution administration

Test Run Queue

No test runs in the queue

Target Boards

#	Model
1	idm
2	idm
1	idm
1	idm
1	idm
1	idm
1	idm
1	idm

Test Automation Framework

0. History Benchmark Functional all both runs

Label tests runs

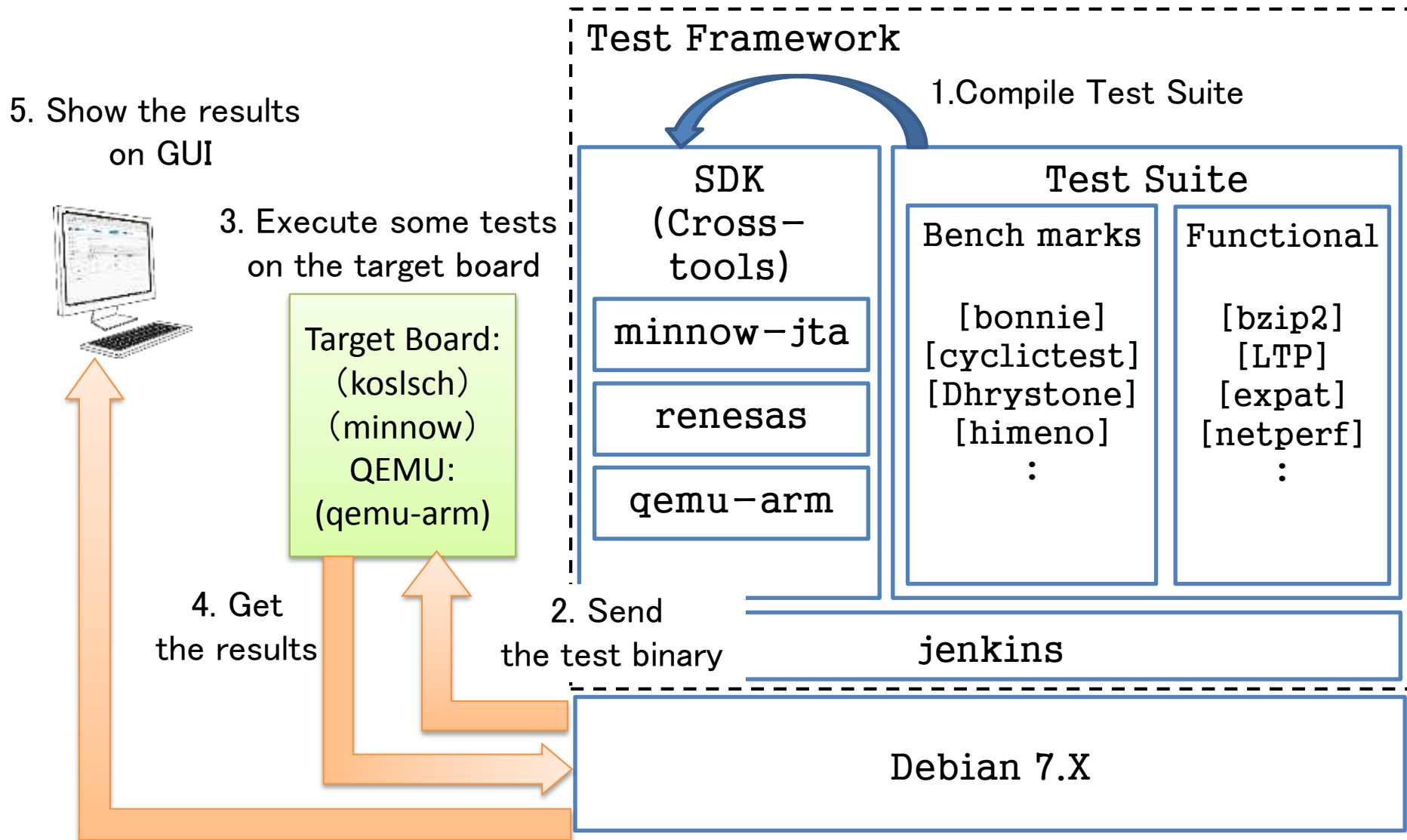
Test	Run	Time	Platform SDK	Device
Functional Unchecked	✓	Sep 9, 2015 8:31:04 PM	Platform SDK 1	req02
Functional path 0	✓	Sep 9, 2015 8:30:56 PM		req02
Functional path 1	✓	Sep 9, 2015 8:30:48 PM		req02
Functional path 2	✓	Sep 9, 2015 8:30:40 PM		req02
Functional path 3	✓	Sep 9, 2015 8:30:32 PM		req02
Functional path 4	✓	Sep 9, 2015 8:30:24 PM		req02
Functional path 5	✓	Sep 9, 2015 8:29:56 PM		req02
Functional path 6	✓	Sep 9, 2015 8:29:48 PM		req02
Functional path 7	✓	Sep 9, 2015 8:29:40 PM		req02
Functional path 8	✓	Sep 9, 2015 8:29:32 PM		req02
Functional path 9	✓	Sep 9, 2015 8:29:24 PM		req02
Functional path 10	✓	Sep 9, 2015 8:29:16 PM		req02
Functional path 11	✓	Sep 9, 2015 8:29:08 PM		req02
Functional path 12	✓	Sep 9, 2015 8:29:00 PM		req02
Functional path 13	✓	Sep 9, 2015 8:28:52 PM		req02
Functional path 14	✓	Sep 9, 2015 8:28:44 PM		req02
Functional path 15	✓	Sep 9, 2015 8:28:36 PM		req02
Functional path 16	✓	Sep 9, 2015 8:28:28 PM		req02
Functional path 17	✓	Sep 9, 2015 8:28:20 PM		req02
Functional path 18	✓	Sep 9, 2015 8:28:12 PM		req02
Functional path 19	✓	Sep 9, 2015 8:28:04 PM		req02
Functional path 20	✓	Sep 9, 2015 8:27:56 PM		req02
Functional path 21	✓	Sep 9, 2015 8:27:48 PM		req02
Functional path 22	✓	Sep 9, 2015 8:27:40 PM		req02
Functional path 23	✓	Sep 9, 2015 8:27:32 PM		req02
Functional path 24	✓	Sep 9, 2015 8:27:24 PM		req02
Functional path 25	✓	Sep 9, 2015 8:27:16 PM		req02
Functional path 26	✓	Sep 9, 2015 8:27:08 PM		req02
Functional path 27	✓	Sep 9, 2015 8:27:00 PM		req02
Functional path 28	✓	Sep 9, 2015 8:26:52 PM		req02
Functional path 29	✓	Sep 9, 2015 8:26:44 PM		req02
Functional path 30	✓	Sep 9, 2015 8:26:36 PM		req02
Functional path 31	✓	Sep 9, 2015 8:26:28 PM		req02
Functional path 32	✓	Sep 9, 2015 8:26:20 PM		req02
Functional path 33	✓	Sep 9, 2015 8:26:12 PM		req02
Functional path 34	✓	Sep 9, 2015 8:26:04 PM		req02
Functional path 35	✓	Sep 9, 2015 8:25:56 PM		req02
Functional path 36	✓	Sep 9, 2015 8:25:48 PM		req02
Functional path 37	✓	Sep 9, 2015 8:25:40 PM		req02
Functional path 38	✓	Sep 9, 2015 8:25:32 PM		req02
Functional path 39	✓	Sep 9, 2015 8:25:24 PM		req02
Functional path 40	✓	Sep 9, 2015 8:25:16 PM		req02
Functional path 41	✓	Sep 9, 2015 8:25:08 PM		req02
Functional path 42	✓	Sep 9, 2015 8:25:00 PM		req02
Functional path 43	✓	Sep 9, 2015 8:24:52 PM		req02
Functional path 44	✓	Sep 9, 2015 8:24:44 PM		req02
Functional path 45	✓	Sep 9, 2015 8:24:36 PM		req02
Functional path 46	✓	Sep 9, 2015 8:24:28 PM		req02
Functional path 47	✓	Sep 9, 2015 8:24:20 PM		req02
Functional path 48	✓	Sep 9, 2015 8:24:12 PM		req02
Functional path 49	✓	Sep 9, 2015 8:24:04 PM		req02
Functional path 50	✓	Sep 9, 2015 8:23:56 PM		req02
Functional path 51	✓	Sep 9, 2015 8:23:48 PM		req02
Functional path 52	✓	Sep 9, 2015 8:23:40 PM		req02
Functional path 53	✓	Sep 9, 2015 8:23:32 PM		req02
Functional path 54	✓	Sep 9, 2015 8:23:24 PM		req02
Functional path 55	✓	Sep 9, 2015 8:23:16 PM		req02
Functional path 56	✓	Sep 9, 2015 8:23:08 PM		req02
Functional path 57	✓	Sep 9, 2015 8:23:00 PM		req02
Functional path 58	✓	Sep 9, 2015 8:22:52 PM		req02
Functional path 59	✓	Sep 9, 2015 8:22:44 PM		req02
Functional path 60	✓	Sep 9, 2015 8:22:36 PM		req02
Functional path 61	✓	Sep 9, 2015 8:22:28 PM		req02
Functional path 62	✓	Sep 9, 2015 8:22:20 PM		req02
Functional path 63	✓	Sep 9, 2015 8:22:12 PM		req02
Functional path 64	✓	Sep 9, 2015 8:22:04 PM		req02
Functional path 65	✓	Sep 9, 2015 8:21:56 PM		req02
Functional path 66	✓	Sep 9, 2015 8:21:48 PM		req02
Functional path 67	✓	Sep 9, 2015 8:21:40 PM		req02
Functional path 68	✓	Sep 9, 2015 8:21:32 PM		req02
Functional path 69	✓	Sep 9, 2015 8:21:24 PM		req02
Functional path 70	✓	Sep 9, 2015 8:21:16 PM		req02
Functional path 71	✓	Sep 9, 2015 8:21:08 PM		req02
Functional path 72	✓	Sep 9, 2015 8:21:00 PM		req02
Functional path 73	✓	Sep 9, 2015 8:20:52 PM		req02
Functional path 74	✓	Sep 9, 2015 8:20:44 PM		req02
Functional path 75	✓	Sep 9, 2015 8:20:36 PM		req02
Functional path 76	✓	Sep 9, 2015 8:20:28 PM		req02
Functional path 77	✓	Sep 9, 2015 8:20:20 PM		req02
Functional path 78	✓	Sep 9, 2015 8:20:12 PM		req02
Functional path 79	✓	Sep 9, 2015 8:20:04 PM		req02
Functional path 80	✓	Sep 9, 2015 8:19:56 PM		req02
Functional path 81	✓	Sep 9, 2015 8:19:48 PM		req02
Functional path 82	✓	Sep 9, 2015 8:19:40 PM		req02
Functional path 83	✓	Sep 9, 2015 8:19:32 PM		req02
Functional path 84	✓	Sep 9, 2015 8:19:24 PM		req02
Functional path 85	✓	Sep 9, 2015 8:19:16 PM		req02
Functional path 86	✓	Sep 9, 2015 8:19:08 PM		req02
Functional path 87	✓	Sep 9, 2015 8:19:00 PM		req02
Functional path 88	✓	Sep 9, 2015 8:18:52 PM		req02
Functional path 89	✓	Sep 9, 2015 8:18:44 PM		req02
Functional path 90	✓	Sep 9, 2015 8:18:36 PM		req02
Functional path 91	✓	Sep 9, 2015 8:18:28 PM		req02
Functional path 92	✓	Sep 9, 2015 8:18:20 PM		req02
Functional path 93	✓	Sep 9, 2015 8:18:12 PM		req02
Functional path 94	✓	Sep 9, 2015 8:18:04 PM		req02
Functional path 95	✓	Sep 9, 2015 8:17:56 PM		req02
Functional path 96	✓	Sep 9, 2015 8:17:48 PM		req02
Functional path 97	✓	Sep 9, 2015 8:17:40 PM		req02
Functional path 98	✓	Sep 9, 2015 8:17:32 PM		req02
Functional path 99	✓	Sep 9, 2015 8:17:24 PM		req02
Functional path 100	✓	Sep 9, 2015 8:17:16 PM		req02

History of Test Results

Target Boards

List of Test name

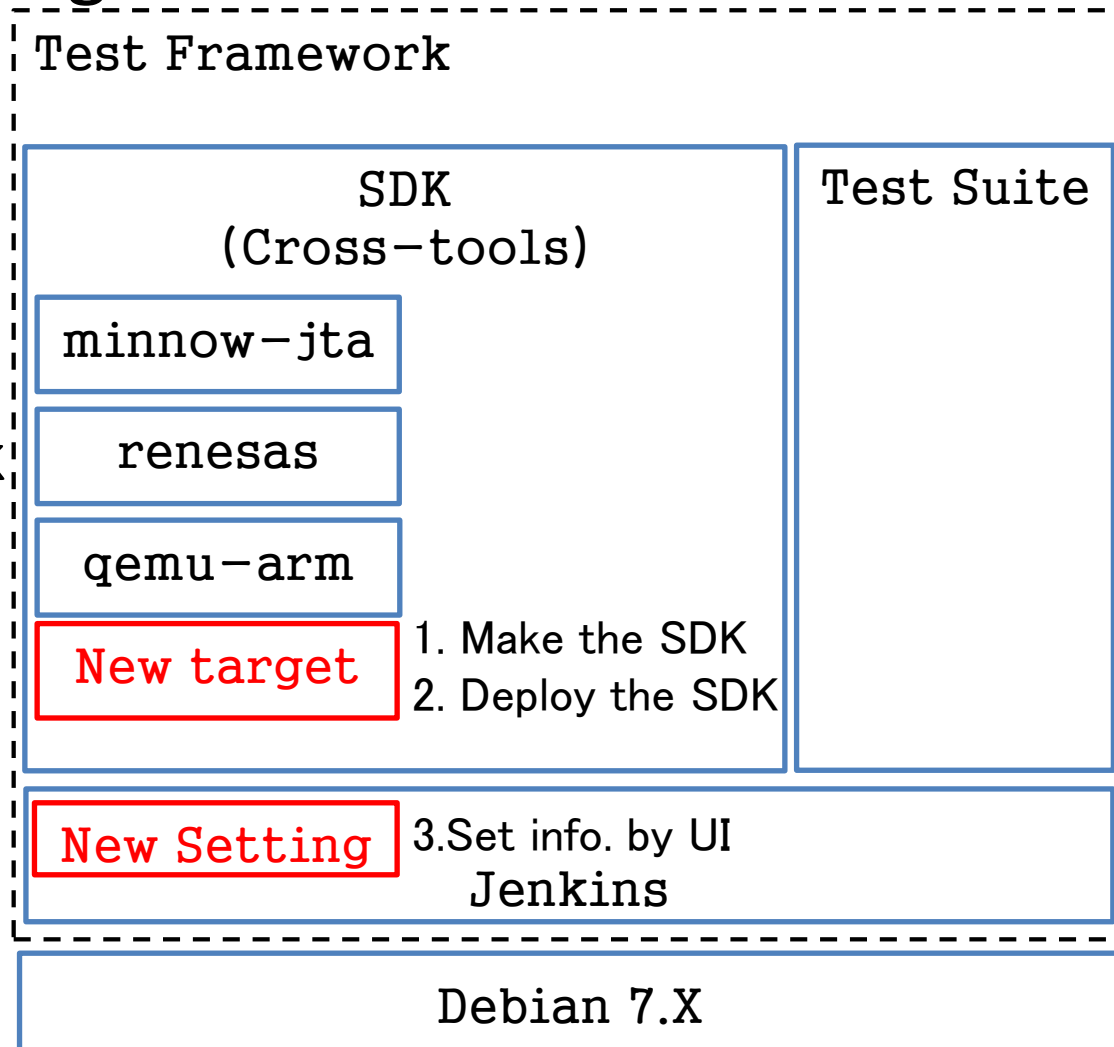
# LTSI Test Environment(Flow)



# How to Customize ?(New Target)

- 3 step to add New Target

- Make the SDK for the target
  - Using yocto project
- Deploy the SDK into Test Framework
- Set target Information by GUI



# How to Customize ?(Raspberry pi 2)

- 3 step to add New Target

- Make the SDK for the target

- Using yocto project

- Deploy the SDK into Test Framework

- Set target Information by GUI

Test Framework

SDK  
(Cross-tools)

minnow-jta

renesas

qemu-arm

New target

1. Make the SDK  
2. Deploy the SDK

Test Suite

New Setting

3. Set info. by UI  
Jenkins

Debian 7.X

# How to Customize ?(Raspberry pi 2)

- Make the SDK

- Getting **poky** from Yocto project

```
$ git clone git://git.yoctoproject.org/poky.git
```

- Getting **meta-raspi** and **meta-jta**

- **meta-raspi** : For making a OS image and SDK for Raspberry pi2

```
$ git clone git://git.yoctoproject.org/meta-raspberrypi
```

- **meta-jta**: For adding Headers and Libs for the Test Suite

```
$ git clone https://bitbucket.org/cogentembedded/meta-jta.git
```



# How to Customize ?(Raspberry pi 2)

- Make the SDK (Cont'd)

- Configure the environment to build

- Execute “**oe-init-build-env**” script in Poky Directory

```
poky$ source oe-init-build-env build-raspi2
```

- then “**build-raspi2**” directory is created like this directory tree.

```
build-raspi2$tree
```

```
└─ conf
```

```
    └─ bblayers.conf
```

```
    └─ local.conf
```

- “build-raspi2” includes a conf directory

- There are “bblayers.conf” and “local.conf” in the conf directory

# How to Customize ?(Raspberry pi 2)

- Make the SDK (Cont'd)

- Setting to build(Cont'd)

- Configure bblayers.conf for meta-raspberrypi & meta-jta

```
BBLAYERS ?= "  
/home/melco/sdk/yocto/poky/meta ¥  
/home/melco/sdk/yocto/poky/meta-yocto ¥  
/home/melco/sdk/yocto/poky/meta-yocto-bsp ¥  
/home/melco/sdk/yocto/poky/meta-raspberrypi ¥  
/home/melco/sdk/yocto/poky/meta-jta ¥
```

Adding the path of  
**"meta-raspberrypi"**  
& **"meta-jta"**

- Configure local.conf for meta-raspberrypi & meta-jta

```
#MACHINE ?= "genericx86-64"  
#MACHINE ?= "mpc8315e-rdb"  
#MACHINE ?= "edgerouter"  
MACHINE ?= "raspberrypi2"  
GPU_MEM = "16"
```

Setting **MACHINE** & **GPU Memory size**  
for raspi2

# How to Customize ?(Raspberry pi 2)

- Make the SDK (Cont'd)
  - Build SDK

Execute "**bitbake meta-toolchain**"  
command in the build-raspi2 Directory

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2$ bitbake meta-toolchain
```

```
Parsing recipes: 100%
```

```
|#####  
Parsing of 912 .bb files complete (0 cached, 912 parsed). 1341 targets, 61 skipped, 0 masked, 0  
errors.
```

```
NOTE: Resolving any missing task queue dependencies
```

```
Build Configuration:
```

```
BB_VERSION      = "1.27.1"
```

```
BUILD_SYS       = "x86_64-linux"
```

```
NATIVELSBSTRING = "Debian-7.8"
```

```
TARGET_SYS      = "arm-poky-linux-gnueabi"
```

```
MACHINE         = "raspberrypi2"
```

```
DISTRO          = "poky"
```

```
DISTRO_VERSION   = "1.8+snapshot-20150908"
```

```
TUNE_FEATURES    = "arm armv7a vfpv4 cortexa7"
```

```
TARGET_FPU       = "vfp-vfpv4-neon"
```

```
meta
```

```
meta-yocto
```

```
meta-yocto-bsp   = "master:c1df471feacaf2590216aa476ce242908dac38cf"
```

```
meta-raspberrypi = "master:17dad9328b100beda1cf870c9075e509b5cbfa90"
```

```
meta-jta        = "master:86387705bfe2ae9495bd661f0c4c7cead8fe06de"
```

To be able to verify "**MACHINE**"  
For raspi2

To be able to verify  
that "**bblayers.conf**" works

# How to Customize ?(Raspberry pi 2)

## • Make the SDK (Cont'd)

### – Build SDK (Cont'd)

- When building SDK finished,  
SDK install script is created at <Build Dir>/tmp/deploy/sdk/

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2$ ls -al tmp/deploy/sdk/
```

```
合計 206104
```

```
drwxr-xr-x 2 melco melco 4096 9月 8 19:04 .
```

```
drwxr-xr-x 5 melco melco 4096 9月 8 14:45 ..
```

```
-rw----- 1 melco melco 9331 9月 8 19:04 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-  
vfpv4-neon-toolchain-1.8+snapshot.host.manifest
```

```
-rwxr-xr-x 1 melco melco 103547364 9月 8 19:04 poky-glibc-x86_64-meta-toolchain-  
cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.sh
```

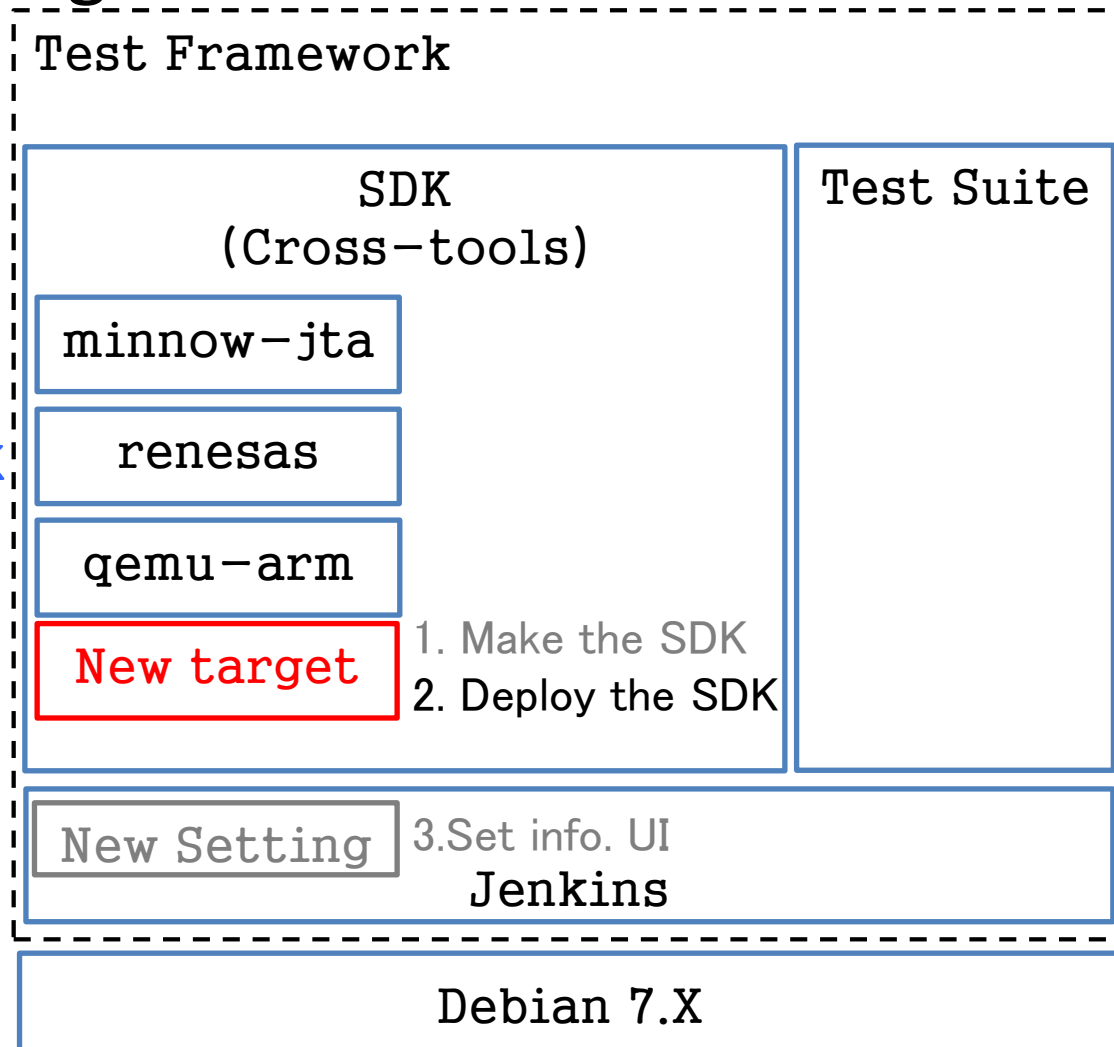
```
-rw----- 1 melco melco 1866 9月 8 19:03 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-  
vfpv4-neon-toolchain-1.8+snapshot.target.manifest
```

This file is the SDK install script.

# How to Customize ?(Raspberry pi 2)

## • 3 step to add New Target

- Make the SDK for the target
  - Using yocto project
- Deploy the SDK into Test Framework
- Set target Information by GUI



# How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework
  - You can Deploy the SDK anywhere
    - This is the default Directory **/home/jenkins/tools/**.  
Minnow, qemu-arm and renesas-arm SDK are already in the directory.

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2/tmp/deploy/sdk$ ./poky-glibc-x86_64-meta-  
toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.sh -y -d /home/jenkins/tools/raspi2  
Poky (Yocto Project Reference Distro) SDK installer version 1.8+snapshot  
=====
```

The directory "/home/jenkins/tools/raspi2" already contains a SDK for this arch.  
If you continue, existing files will be overwritten! Proceed[y/N]? Y

[sudo] password for melco:

Extracting SDK...done

Setting it up...done

SDK has been successfully set up and is ready to be used.

Each time you wish to use the SDK in a new shell session, you need to source the environment  
setup script e.g.

Selecting installing directory and run SDK install script.

# How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)
  - how to Set the Test Framework for the SDK
    - Adding raspi2 configuration on /home/Jenkins/scripts/tools.sh
    - The Test Framework already includes here minnow, qemu-arm and renesas-arm configurations.

```
elif [ "${PLATFORM}" = "raspi2" ];  
then
```

Setting raspi2

“SDKROOT” is the path of the sysroot that there is in deploying the SDK Directory.

```
SDKROOT=$JTA_ENGINE_PATH/tools/raspi2/sysroots/cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi  
# environment script changes PATH in the way that python uses libs from sysroot which is  
not what we want, so save it and use later
```

```
ORIG_PATH=$PATH
```

Setting “PREFIX” for cross compile

```
PREFIX=arm-poky-linux-gnueabi
```

```
source $JTA_ENGINE_PATH/tools/raspi2/environment-setup-cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi
```

Setting this path written the file of environment variable.

```
HOST=arm-poky-linux-gnueabi
```

This file is in the directory deploying the SDK. .

```
unset PYTHONHOME  
env -u PYTHONHOME
```

Set “HOST” for cross compile like “PREFIX”

# How to Customize ?(Raspberry pi 2)

---

- Deploy the SDK into Test Framework(conf.)
  - Set of Test Framework for the Target(raspi2)
    - Adding raspi2 target board configuration on  
/home/jenkins/overlays/boards/<targetname>.board
    - A Sample target board configuration file is template-dev.board
    - When you add a new board,  
you should use template-dev.board



# How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)

```
inherit "base-board"
include "base-params"

IPADDR="set_ip_here"
LOGIN="root"
JTA_HOME="/home/a"
PASSWORD=""
PLATFORM="set platform here (see tools.sh)"
TRANSPORT="ssh"
ARCHITECTURE="set_ia32_or_arm_here"
SATA_DEV="/dev/sdb1"
SATA_MP="/mnt/sata"
USB_DEV="/dev/sda1"
USB_MP="/mnt/usb"
MMC_DEV="/dev/mmcblk0p2"
MMC_MP="/mnt/mmc"

LTP_OPEN_POSIX_SUBTEST_COUNT_POS="1319"
LTP_OPEN_POSIX_SUBTEST_COUNT_NEG="169"
EXPAT_SUBTEST_COUNT_POS="1769"
EXPAT_SUBTEST_COUNT_NEG="41"
```

Setting IP address of a target

Login user name

Directory to run some test

LOGIN user password

Setting Platform name  
elif [ "\${PLATFORM}" = "raspi2" ];

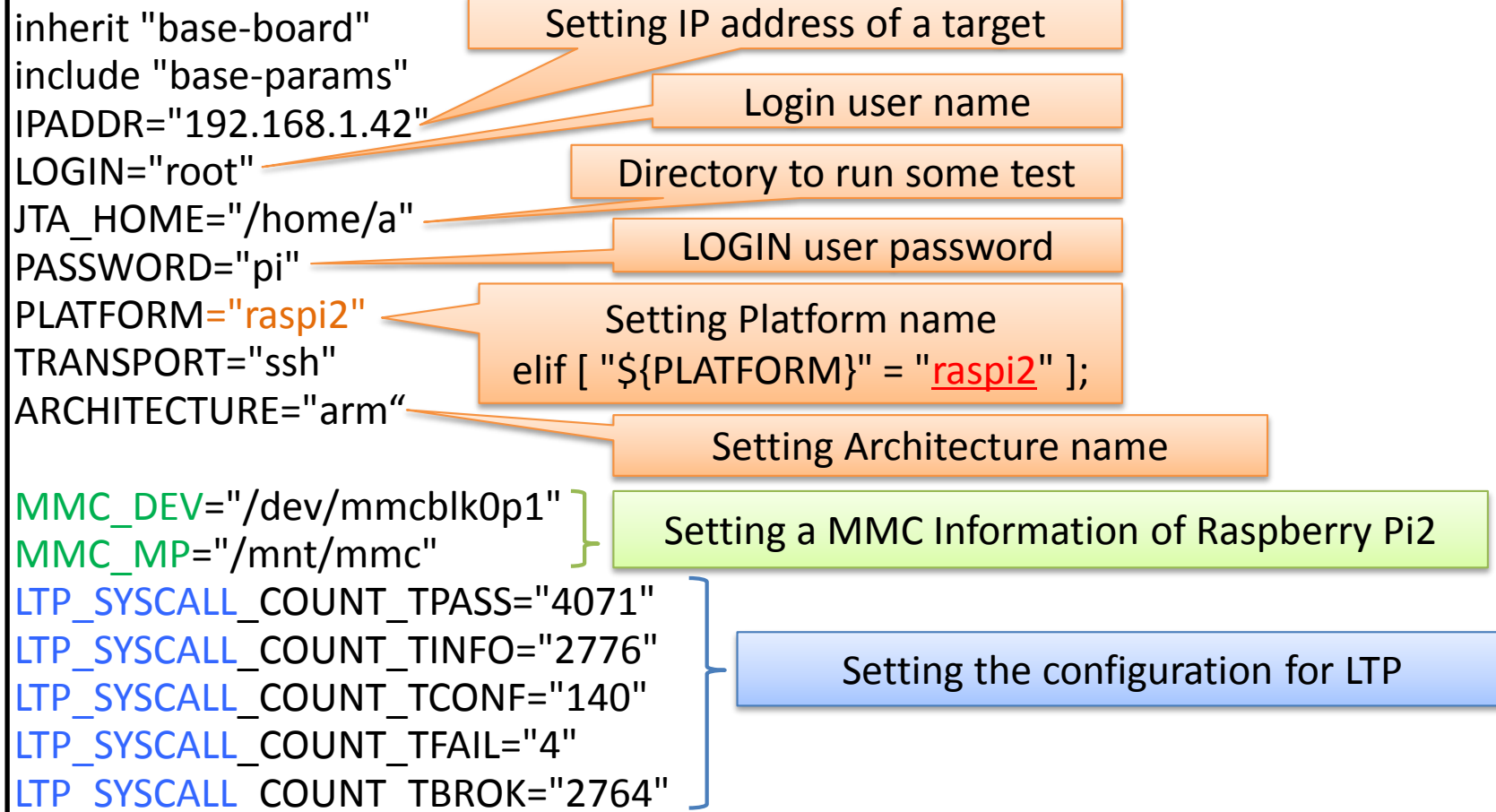
Setting Architecture name

Setting a device Information of the target board  
like SATA, USB and MMC etc.

Setting configuration  
For each test  
like LTP and EXPAT etc.

# How to Customize ?(Raspberry pi 2)

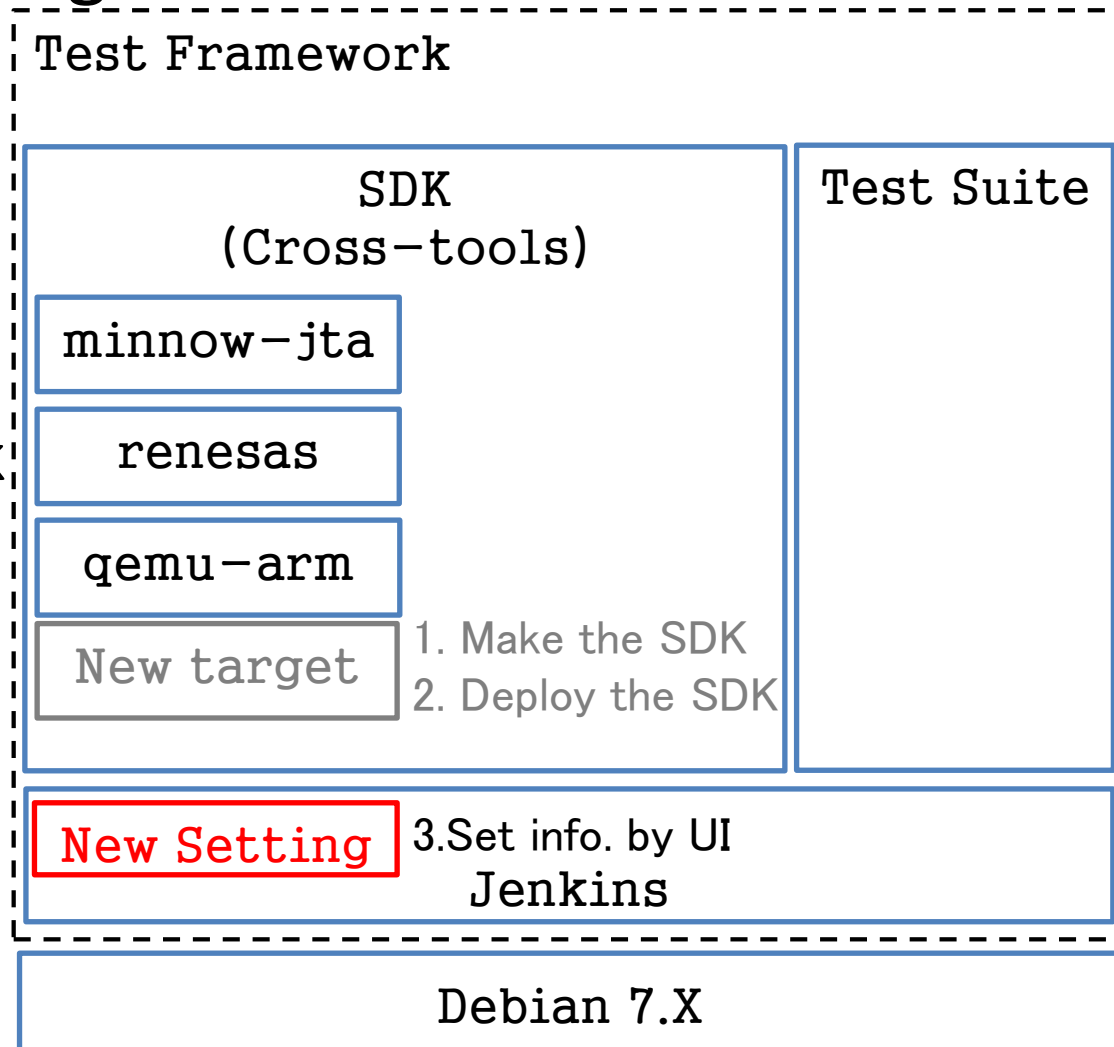
- Deploy the SDK into Test Framework(conf.)
  - For example , <target name>.board for Raspi2



# How to Customize ?(Raspberry pi 2)

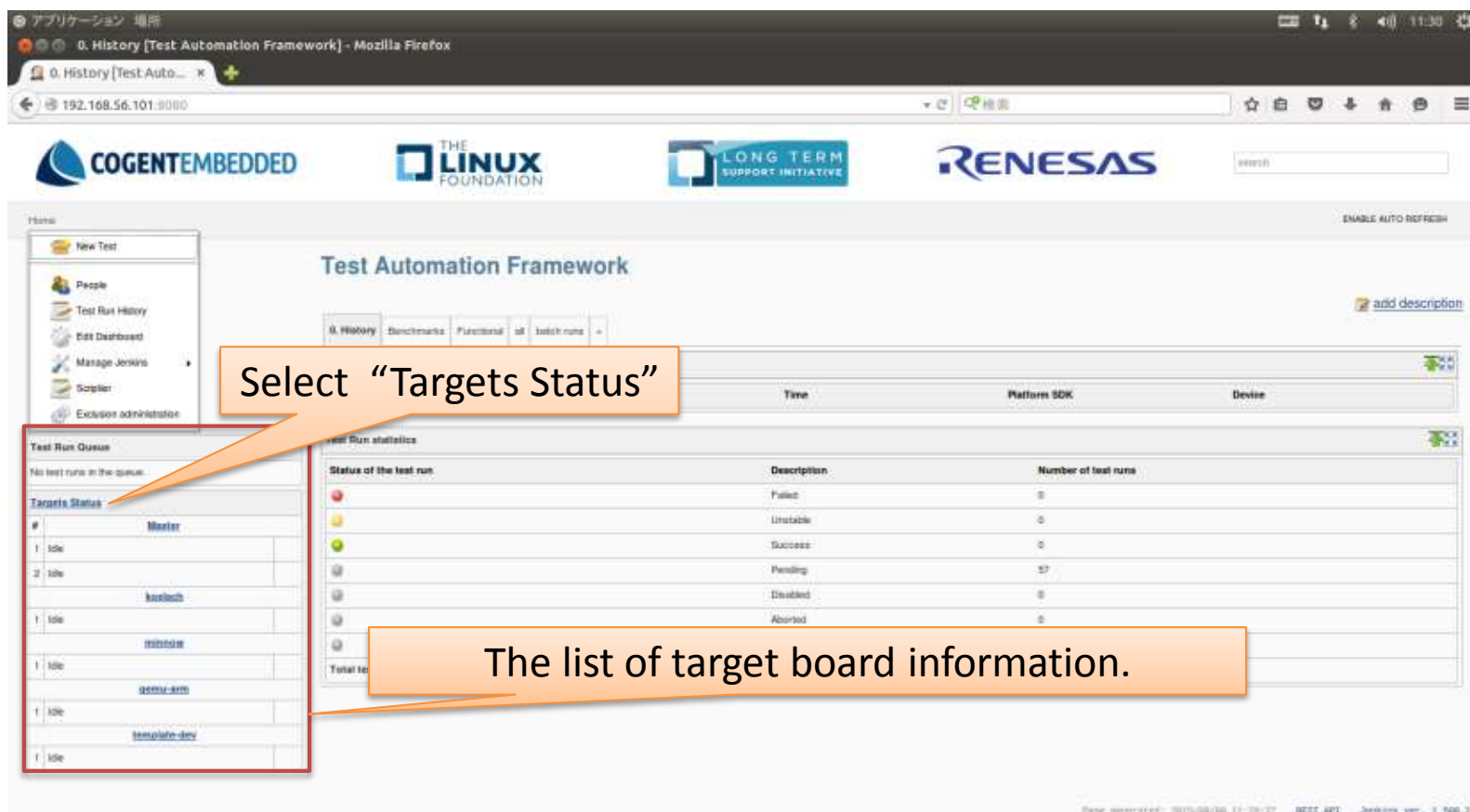
- 3 step to add New Target

- Make the SDK for target
  - Using yocto project
- Deploy the SDK into Test Framework
- Set target information by GUI



# How to Customize ?(Raspberry pi 2)

- Set target Information by UI
  - Select “Targets Status” on top screen of Test Framework



Test Automation Framework

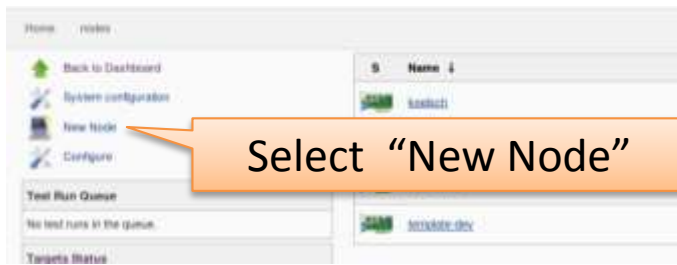
Select “Targets Status”

The list of target board information.

Status of the test run	Description	Number of test runs
Failed		0
Unstable		0
Success		0
Pending		57
Disabled		0
Aborted		0
Total		

# How to Customize ?(Raspberry pi 2)

- Set target Information by UI(conf.)
  - Select “New Node”

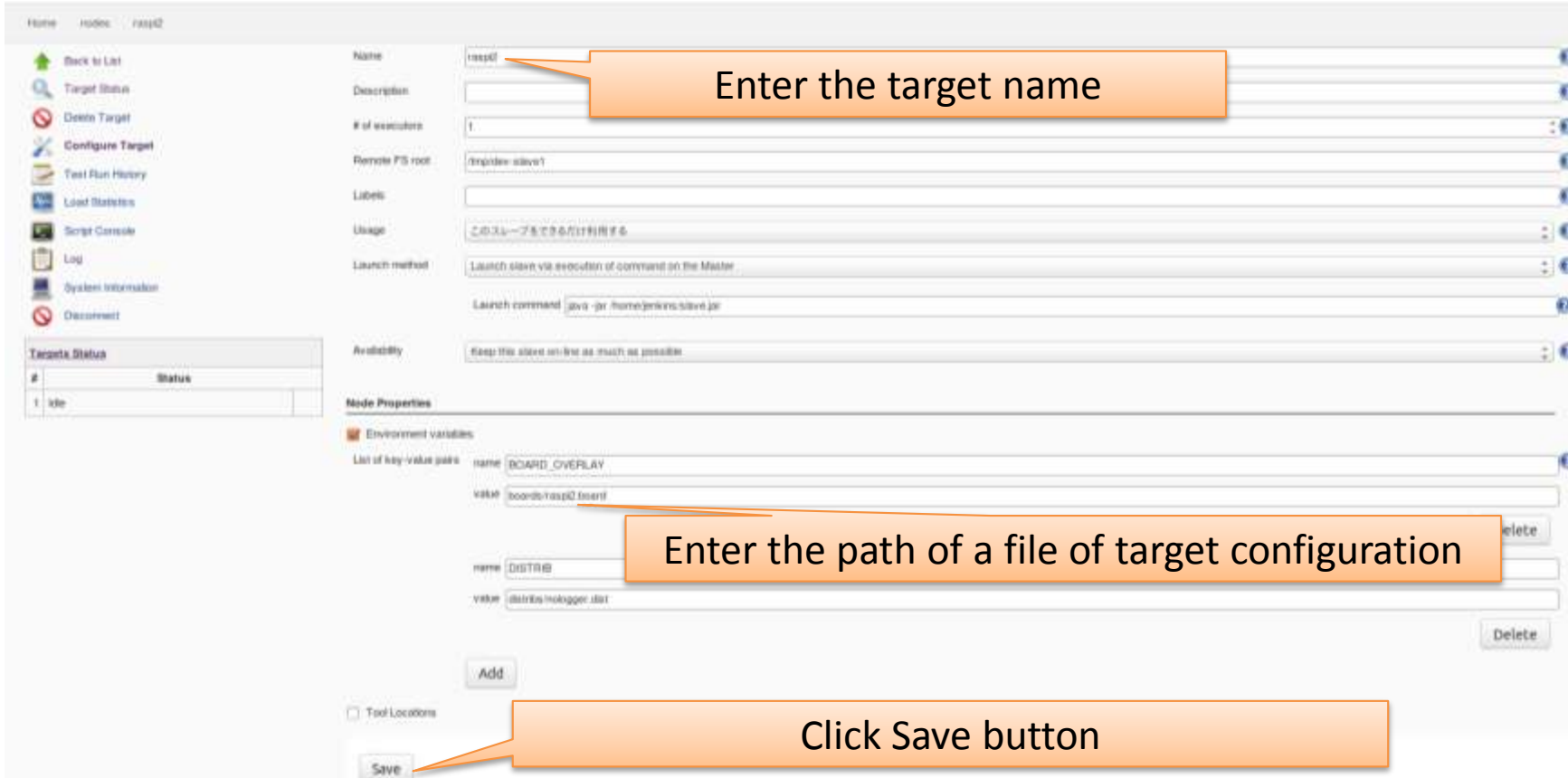


- Then, you can see a configuration form



# How to Customize ?(Raspberry pi 2)

- Set target Information by UI(conf.)
  - You enter just 2 forms  
as “Name” and “List of Key-values pairs”



Home nodes **raspi2**

Back to List Target Status Delete Target Configure Target Test Run History Load Statistics Script Console Log System Information Disconnect

**Targets Status**

#	Status
1	idle

Name: **raspi2** Enter the target name

Description:

# of executors: 1

Remote FS root: /tmp/idee-slave1

Labels:

Usage: このエディタで利用できるだけ利用する

Launch method: Launch slave via execution of command on the Master

Launch command: java -jar /home/perkins/slaved.jar

Availability: Keep this slave on-line as much as possible

**Node Properties**

Environment variables

List of key-value pairs

name	value	delete
BOARD_OVERLAY	/boards/raspi2/bsm1f	delete
DISTRIB	/distrib/mokkugei.dat	delete

Add

Tool Locations

Save Click Save button

# How to Customize ?(Raspberry pi 2)

- Set target Information by UI(conf.)
  - You can see a target list that New target board was added

Test Run Queue		
No test runs in the queue.		
Targets Status		
#	Master	
1	Idle	
2	Idle	
koelsch		
1	Idle	
minnow		
1	Idle	
gemu-arm		
1	Idle	
raspi2		
1	Idle	
template-dev		
1	Idle	

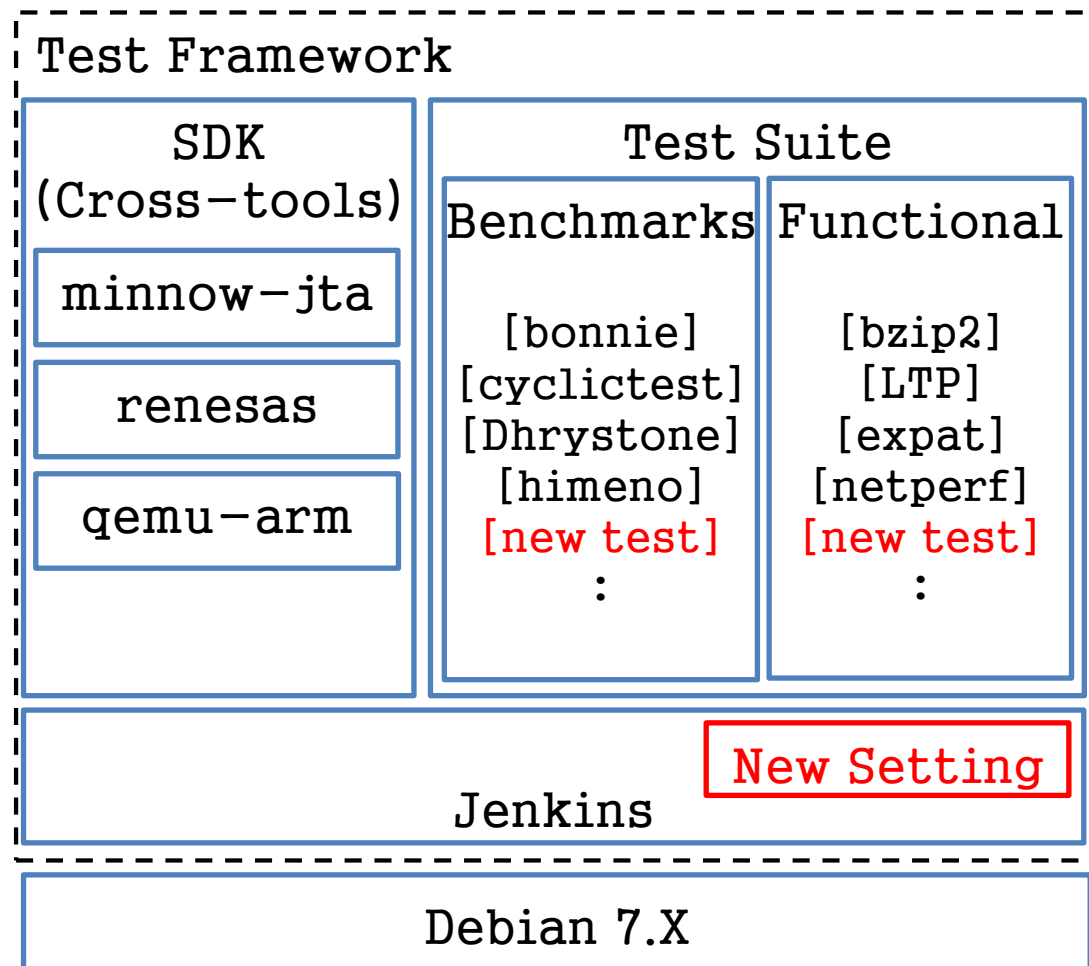
New target name is raspi2

Finish adding new target as Raspberry pi 2!!!

# How to Customize ?(New Test Suite)

## • 3 step to add New Test Suite

- Create a script for running a new test suite
- Deploy the script and a test suite tarball
- Set the test suite information by GUI

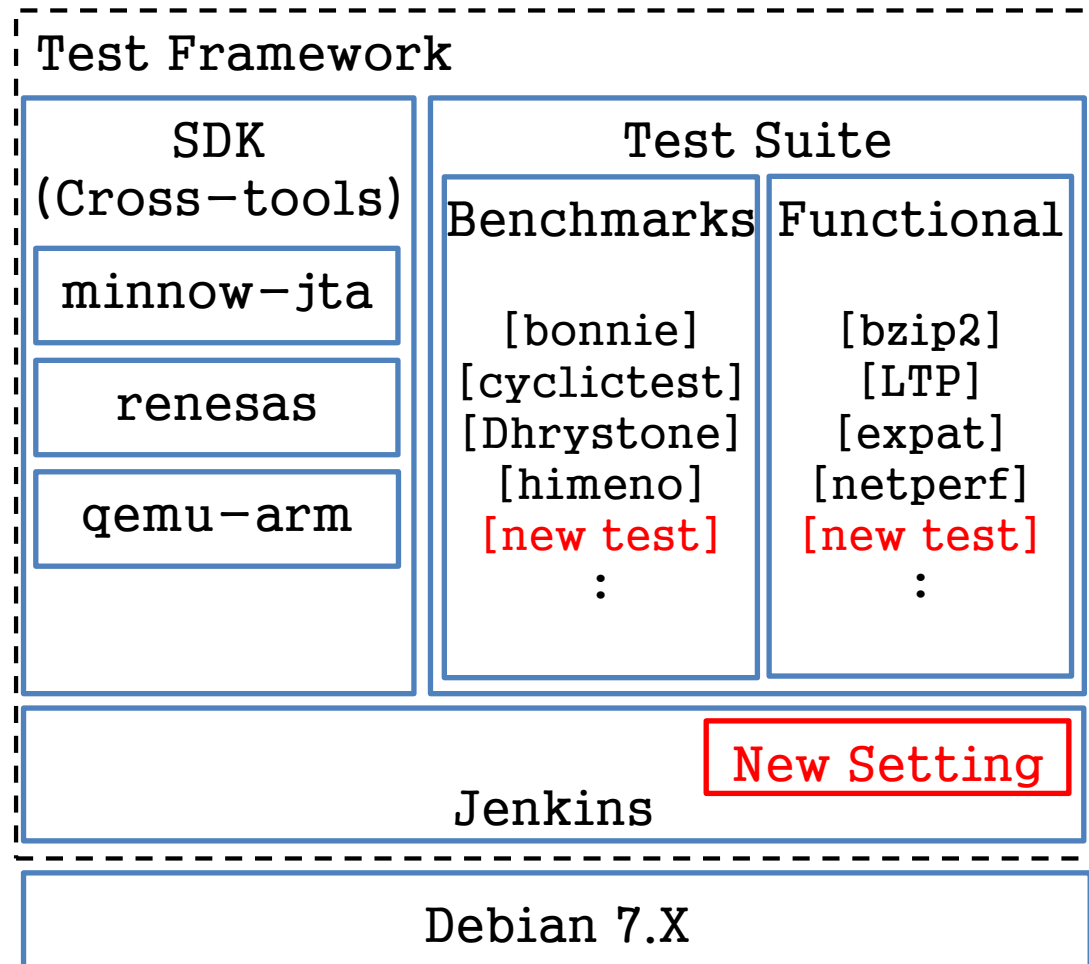




# How to Customize ?(New Test Suite)

## • 3 step to add New Test Suite

- Create a script for running a new test suite
- Deploy the script and a test suite tarball
- Set the test suite information by GUI



# How to Customize ?(Linux Test Project)

- Create the script named “ltp-all.sh” (1)

```
tarball=ltp-full-20150420.tar.bz2
```

To describe tarball name of the adding test suite

```
function test_build {
```

To describe procedure of creating test module using cross compile.

```
    make autotools  
    ./configure CC="${CC}" AR="${AR}" RANLIB="${RANLIB}" LDFLAGS="${LDFLAGS}" --  
without-perl --without-python --target=$PREFIX --host=$PREFIX --  
prefix=`pwd`/target_bin --build=`uname -m`-unknown-linux-gnu  
    make CC="${CC}"  
    make install  
}
```

To describe procedure of deploying the test module to the target.

```
function test_deploy {
```

```
    put -r target_bin /tmp/jta.$TESTDIR/  
}
```

To describe commands to execute the test module on target.

```
function test_run {
```

```
    safe_cmd "cd /tmp/jta.$TESTDIR/target_bin; ./runltp -f syscalls |  
tee $JTA_HOME/jta.$TESTDIR/$TESTDIR.log"  
}
```

In this case, to show running LTP command and collecting the result log.

# How to Customize ?(Linux Test Project)

## • Create the script named “ltp-all.sh” (conf.)

```
function test_processing {
## To judge test result
    assert_define LTP_SYSCALL_COUNT_TPASS
    assert_define LTP_SYSCALL_COUNT_TINFO
    assert_define LTP_SYSCALL_COUNT_TCONF
    assert_define LTP_SYSCALL_COUNT_TFAIL
    assert_define LTP_SYSCALL_COUNT_TBROK

    TPASS_CRIT="TPASS : "
    TINFO_CRIT="TINFO : "
    TCONF_CRIT="TCONF : "
    TFAIL_CRIT="TFAIL : "
    TBROK_CRIT="TBROK : "

    log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TPASS "${TPASS_CRIT}" "TPASS"
    log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TINFO "${TINFO_CRIT}" "TINFO"
    log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TCONF "${TCONF_CRIT}" "TCONF"
    log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TFAIL "${TFAIL_CRIT}" "TFAIL"
    log_compare "$TESTDIR" $LTP_SYSCALL_COUNT_TBROK "${TBROK_CRIT}" "TBROK"

    echo "test_processing done"
}
. $JTA_ENGINE_PATH/scripts/functional.sh
```

To describe judgment and analysis process of test results

Verify definitions

To define Keywords to search in the log

To compare definitions and result log

Define on “<target name>.board”

ltp-all.sh is inherited functional.sh  
The above functions are called by it.

# How to Customize ?(Linux Test Project)

- Functional.sh inherits from ltp-all.sh.
  - “functional.sh” is defined on LTSI test by default.

```
source $JTA_ENGINE_PATH/scripts/overlays.sh
set_overlay_vars

source $JTA_ENGINE_PATH/scripts/reports.sh
source $JTA_ENGINE_PATH/scripts/functions.sh
```

To include common scripts and execute overlay using Test plan and spec files. Test plan and Spec files provide the very flexibility in configuring tests to be run on different boards and scenarios in the Test Framework.

```
pre_test $TESTDIR

if $Rebuild; then
    build
fi

deploy

test_run

get_testlog $TESTDIR

test_processing
```

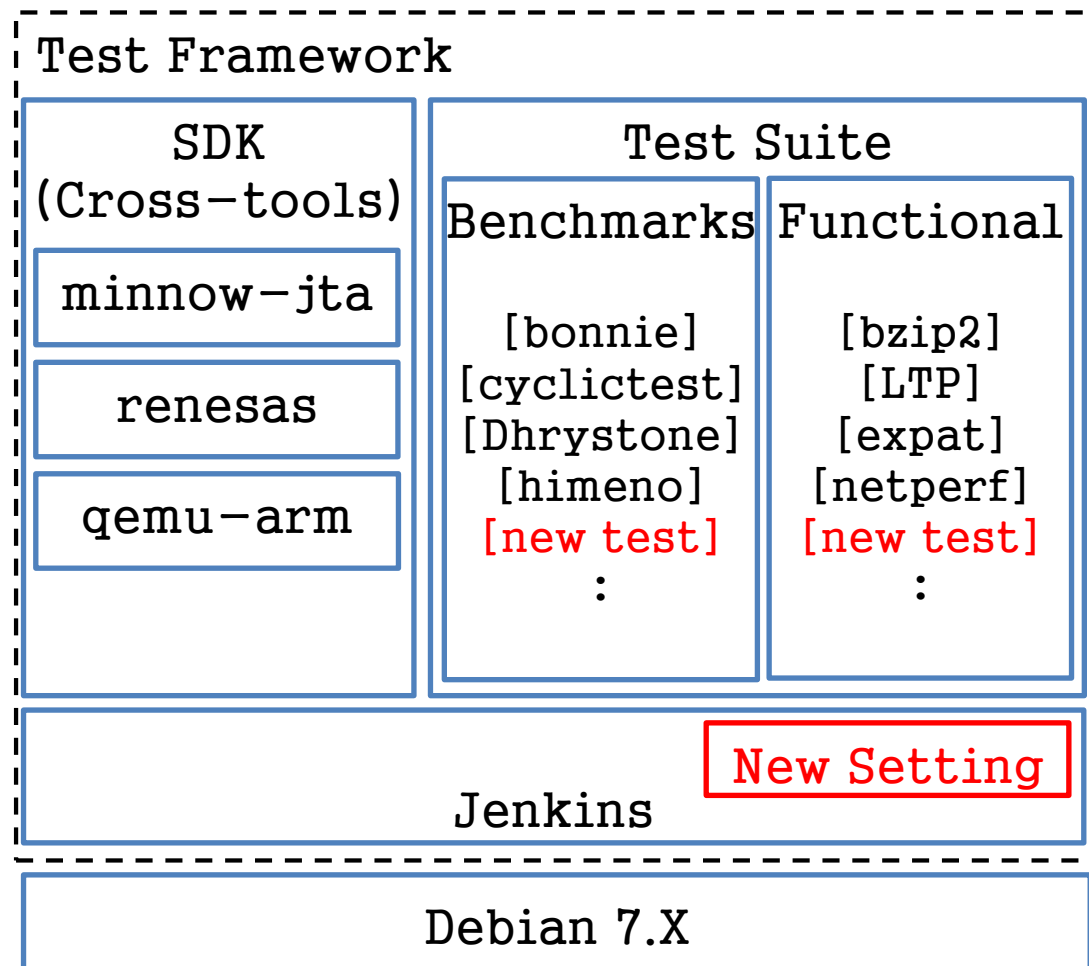
Standard sequence for running test script on the Test Framework.

- “Pre\_test” is checking precondition like connection with target board and target version and so on before running test.
- “Build” is executing test\_build function which is defined on “ltp-all.sh” as I explained.
- “Deploy” is executing test\_deploy function.
- “Get\_testlog” is getting the executing log after running the test..
- “test\_run” and “test\_processing” are defined on “ltp-all.sh”.

# How to Customize ?(New Test Suite)

## • 3 step to add New Test Suite

- Create a script for running a new test suite
- Deploy the script and a test suite tarball
- Set the test suite information by GUI



# How to Customize ?(Linux Test Project)

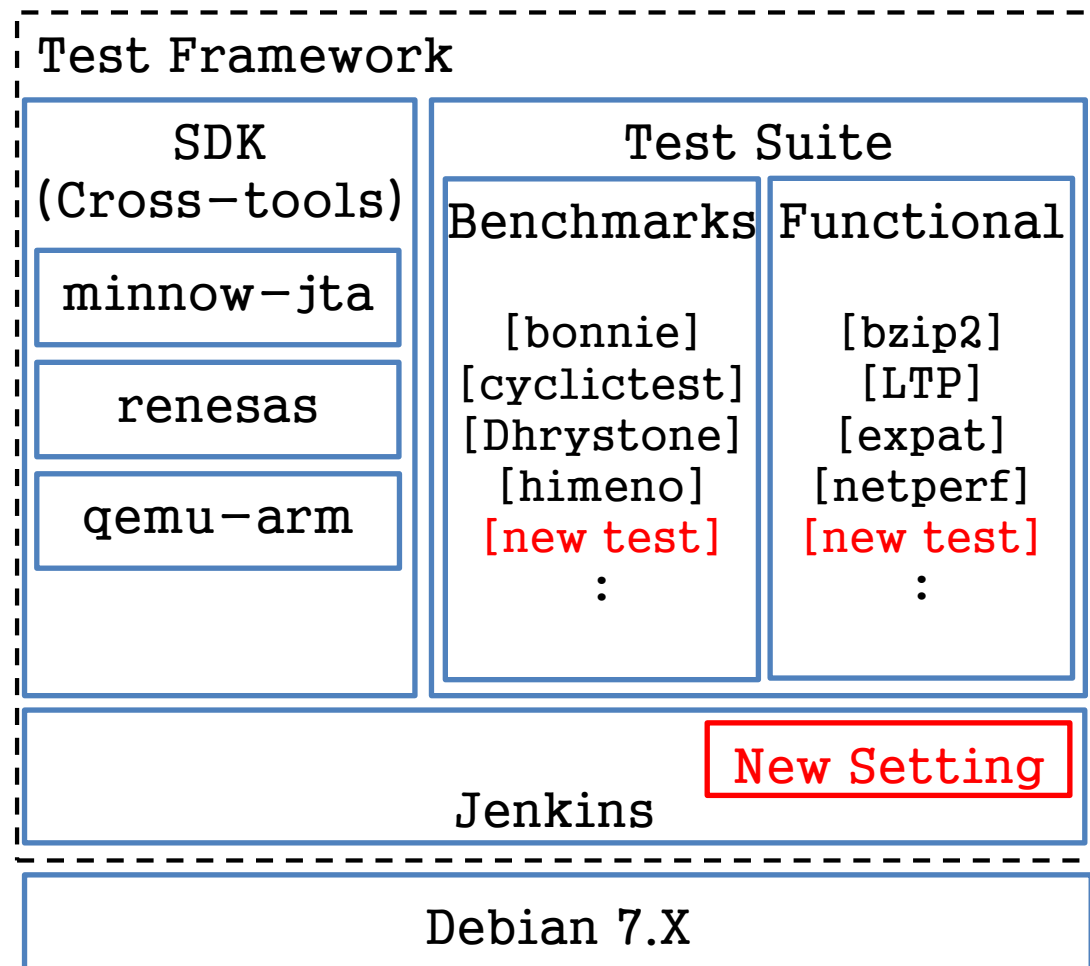
- Deploy the script and test suite tarball
  - To create work directory “Functional.LTP.all” under /home/jenkins/tests/.
    - Arbitrary directory name can be used but the above is standard.
  - To obtain tarball of LTP from the below site
    - <https://github.com/linux-test-project/ltptest/releases/tag/20150420>
  - To put the created script and tarball under Functional.LTP.all.

```
melco@debian-7:/home/jenkins/tests/ Functional.LTP.all$ ls  
ltptest-all.sh ltptest-full-20150420.tar.bz2
```

# How to Customize ?(New Test Suite)

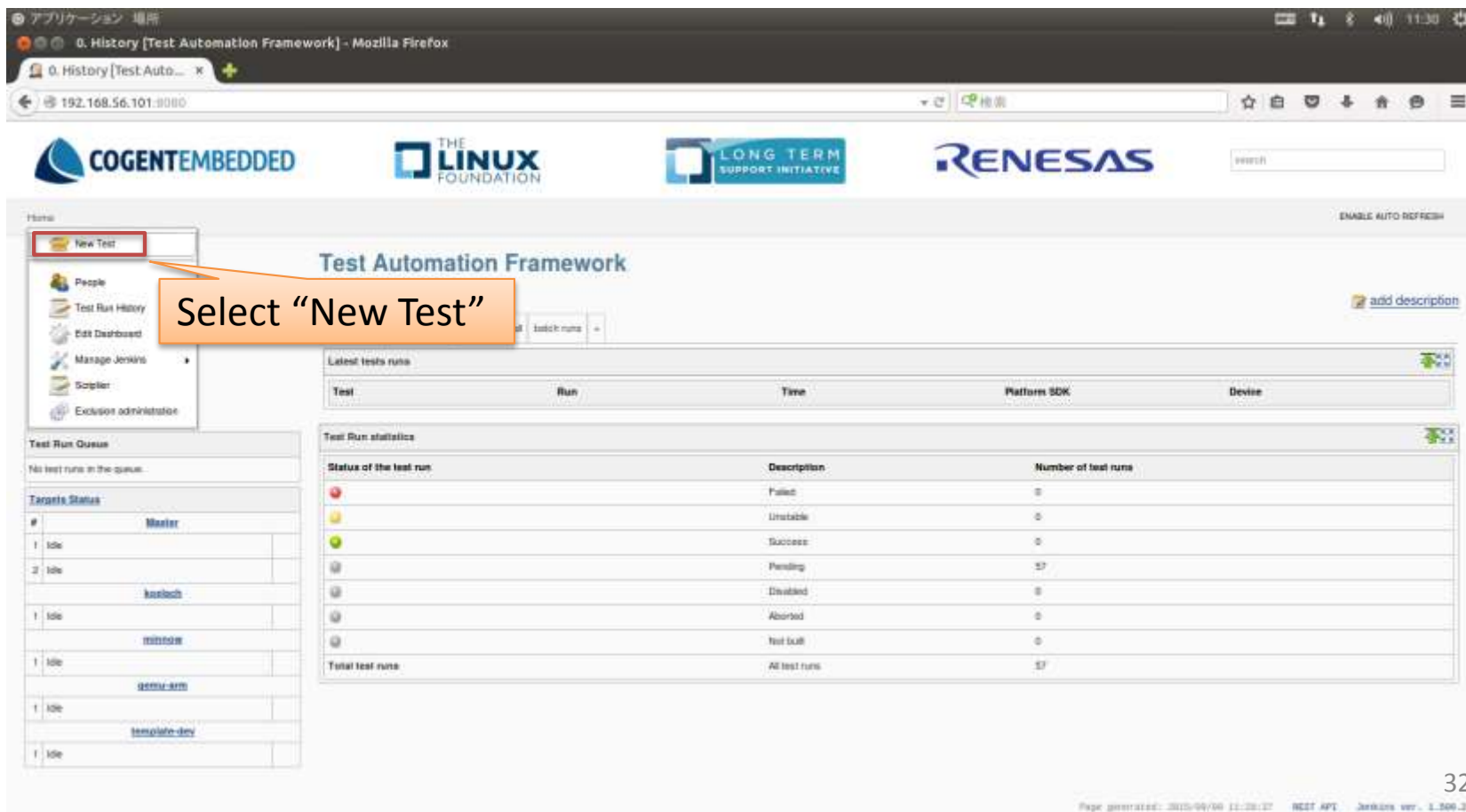
## • 3 step to add New Test Suite

- Create a script for running a new test suite
- Deploy the script and a test suite tarball
- Set the test suite information by GUI



# How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - To select “New Test” on the left side of screen of Test Framework



Test Automation Framework

Latest tests runs

Test	Run	Time	Platform SDK	Device

Test Run statistics

Status of the test run	Description	Number of test runs
Failed	Failed	0
Unstable	Unstable	0
Success	Success	0
Pending	Pending	17
Disabled	Disabled	0
Aborted	Aborted	0
Test fail	Test fail	0
Total test runs	All test runs	17

Test Run Queue

No test runs in the queue.

Targets Status

#	Name	Status
1	Master	
2	kunit4ch	
1	mips64le	
1	gemu-arm	
1	gemu64le-dev	
1		



# How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - To input Test name
  - To choose “Copy existing Test” and Copy from

Home 0. History

**New Test**

People

Test Run History

Edit Dashboard

Manage Jenkins

Scriptler

Exclusion administration

**Test Run Queue**

No test runs in the queue.

**Targets Status**

#	Master
1	Idle
2	Idle

[koelsch](#)

1	Idle
---	------

Test name

☐ Inheritance Project  
This is a project that allows basic inheritance of properties to occur. It is broadly compatible with the free-style-project class for most purposes, b

☐ Test a free-style software project  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for s

☐ Build a maven2/3 project  
Build a maven 2/3 project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

☐ Monitor an external test  
This type of job allows yo [documentation for more d](#)

☐ Test run multi-configuration project  
Suitable for projects that need a large n [in configurations, such as testing on multiple environments, platform-specific builds, etc.](#)

☒ Copy existing Test  
Copy from

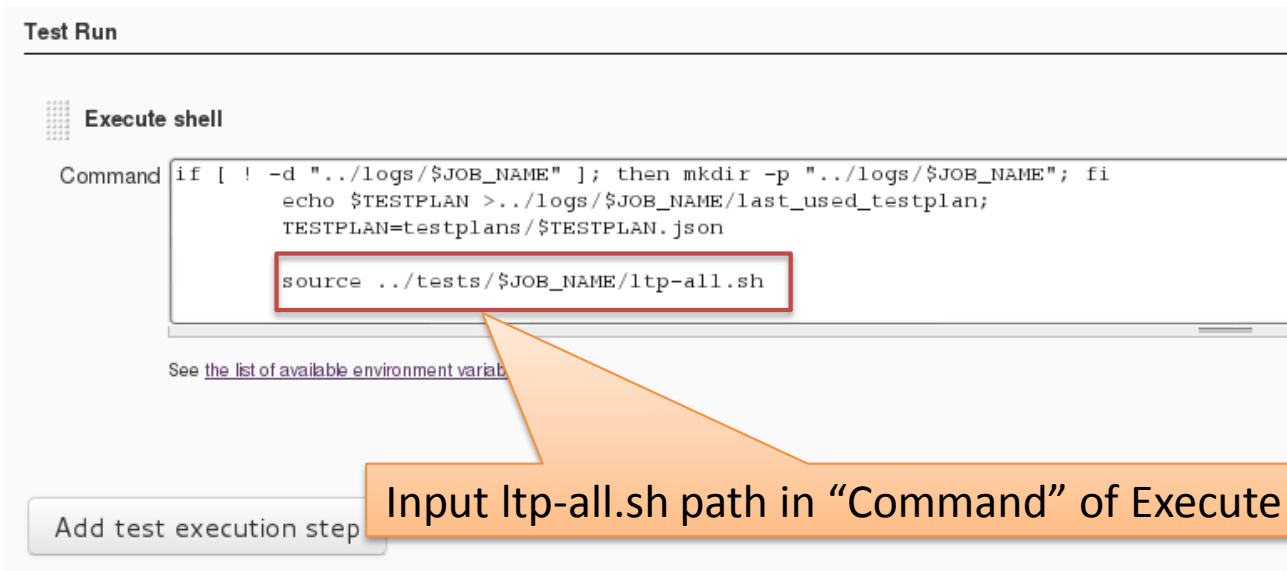
OK

Input Funtional.LTP.all in “Test name”

Select “Copy existing Test” and input Functional.LTP.Open\_Posix in “Copy from”

# How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - To input the created script path in “Command” field of “Execute shell” of “Test Run”



**Test Run**

**Execute shell**

Command

```
if [ ! -d "../logs/$JOB_NAME" ]; then mkdir -p "../logs/$JOB_NAME"; fi
echo $TESTPLAN >../logs/$JOB_NAME/last_used_testplan;
TESTPLAN=testplans/$TESTPLAN.json
source ../tests/$JOB_NAME/ltp-all.sh
```

See [the list of available environment variables](#)

Add test execution step

Input ltp-all.sh path in “Command” of Execute shell

# How to Customize ?(Linux Test Project)

- Set test suite Information by GUI
  - You can see new test suite name in Functional Tab

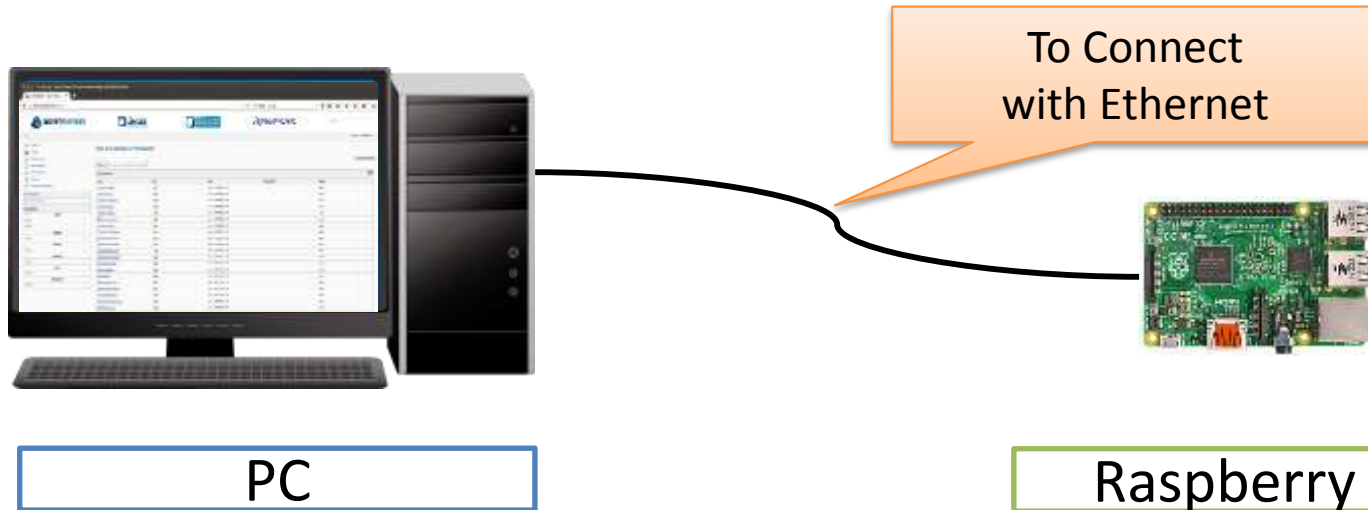


New test suite

Finish adding new test suite as Linux Test Project !!!

# Run new LTP on Raspberry Pi2

## • Test Environment



- Running Test Framework on Debian7.8
- Show the test result by web browser
- Adding the SDK Raspberry pi2
- Adding the LTP-20150420

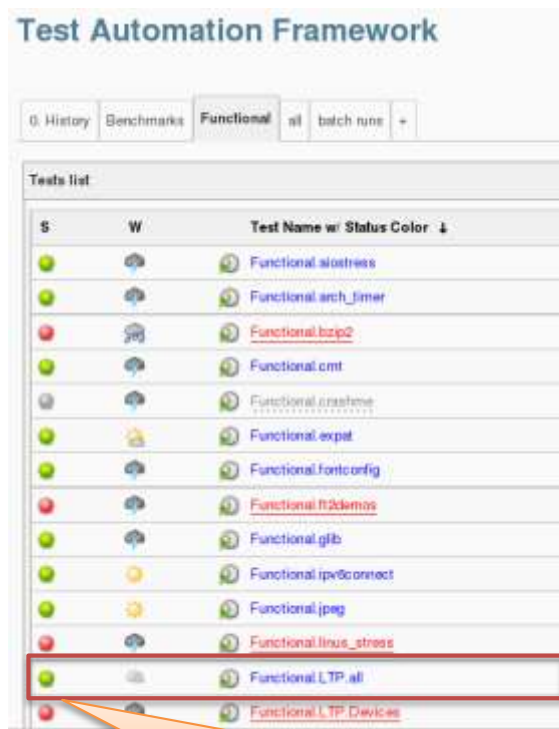
- Running poky 1.8 from Yocto project
- Kernel version: 3.18

# Run new LTP on Raspberry Pi2 (Cont'd)

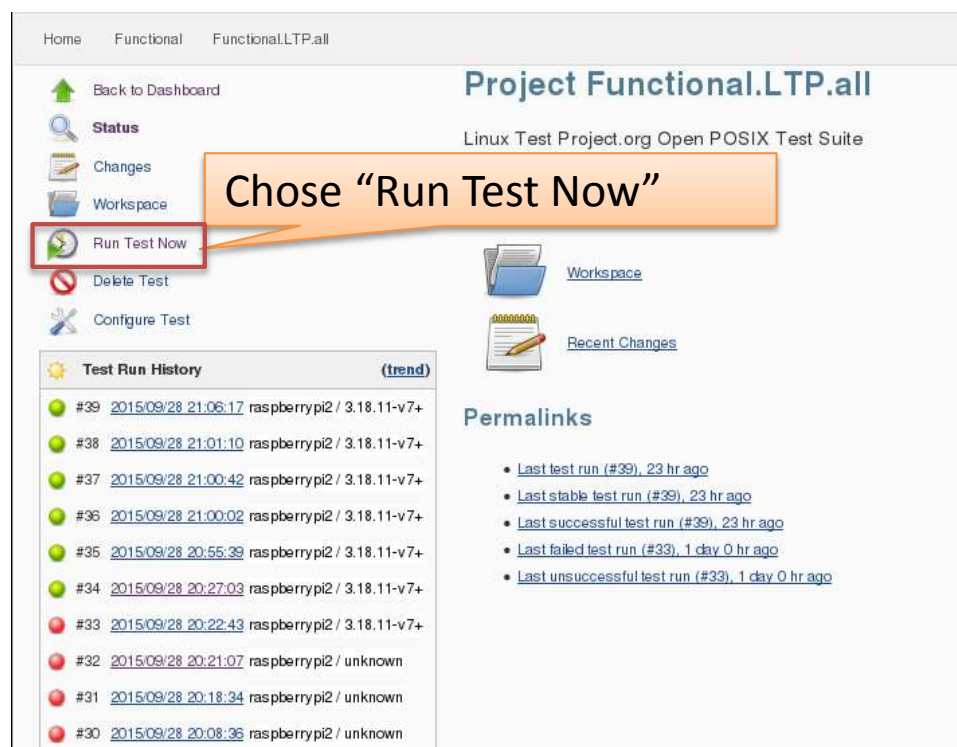
- To select LTP-20150420

- To choose  
Functional.LTP.all

- To choose “Run Test Now”

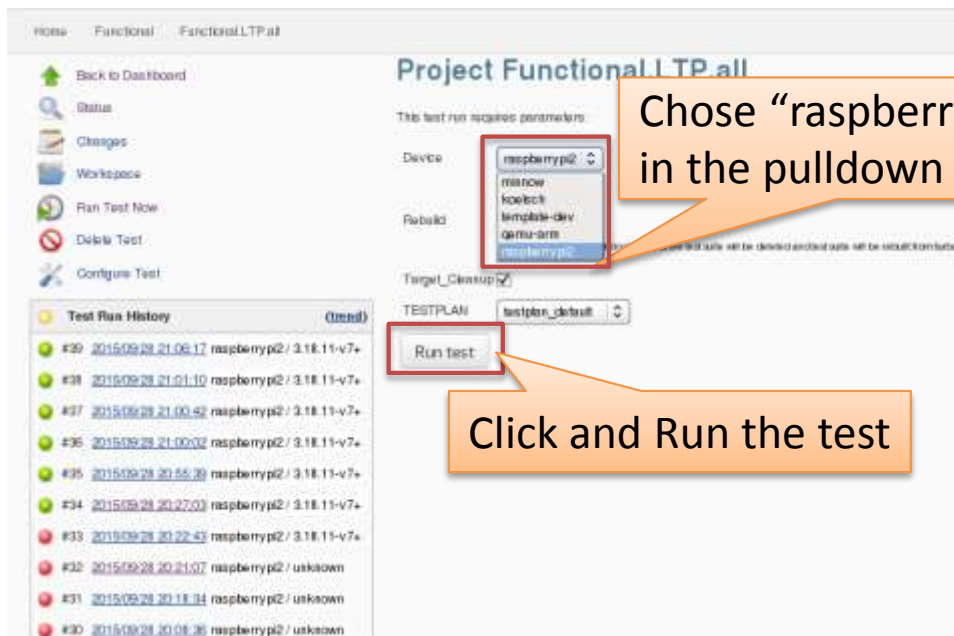


Chose “Functional.LTP.all”



# Run new LTP on Raspberry Pi2 (Cont'd)

- To Run LTP-20150420

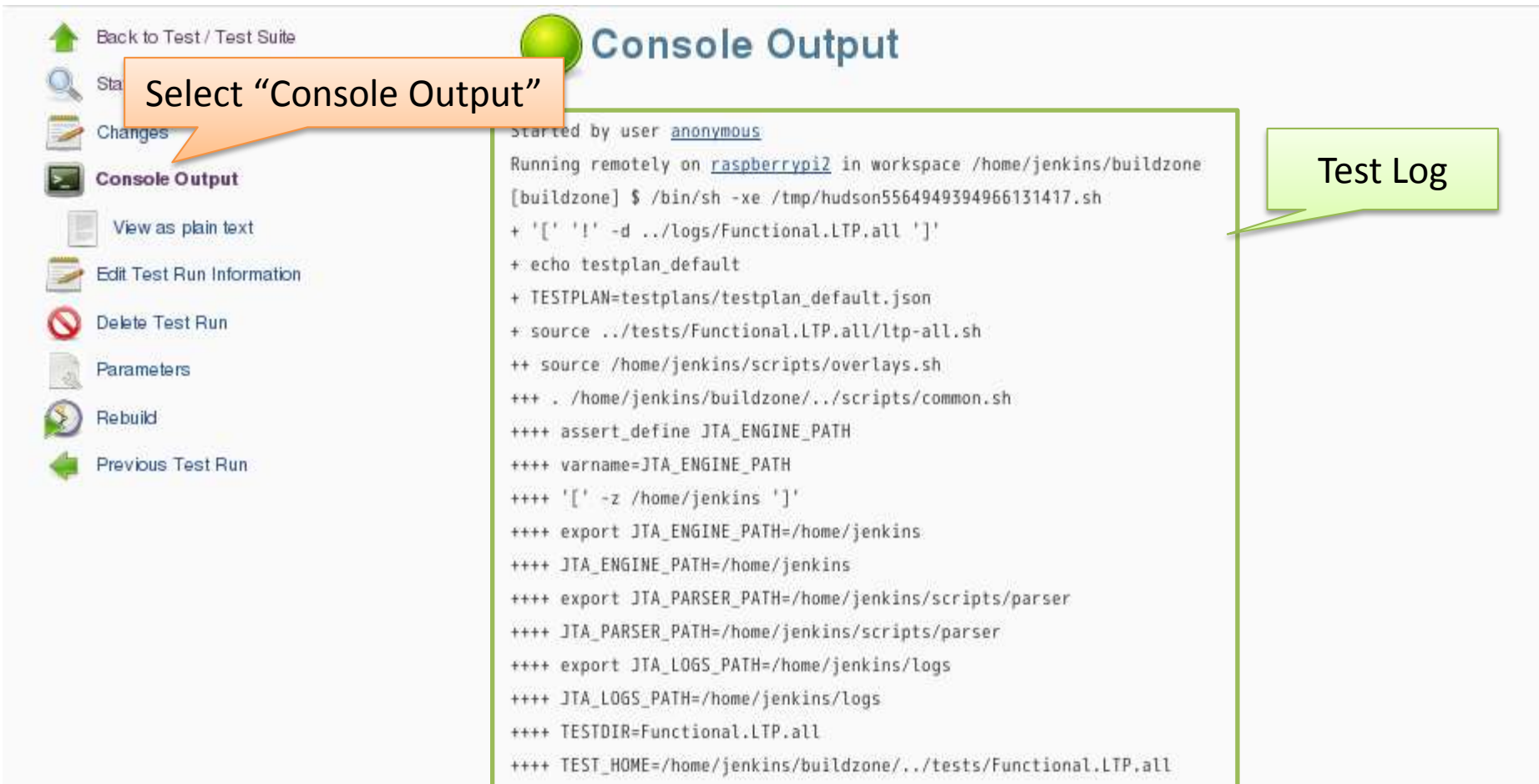


The screenshot shows the 'Project Functional LTP all' interface. On the left is a sidebar with navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Run Test Now', 'Delete Test', and 'Configure Test'. Below this is a 'Test Run History' table with columns for ID, timestamp, device, and version. The table lists several test runs, some successful (green) and some failed (red). The main area on the right is titled 'Project Functional LTP all' and contains configuration options: 'Device' (a dropdown menu with 'raspberrypi2' selected), 'Rebuild', 'Target\_Cleanup' (checked), and 'TESTPLAN' (a dropdown menu with 'testplan\_default' selected). A 'Run test' button is located below these options. Two orange callout boxes provide instructions: one points to the 'Device' dropdown with the text 'Chose "raspberrypi2" in the pulldown', and another points to the 'Run test' button with the text 'Click and Run the test'.

ID	Timestamp	Device	Version
#39	2015/09/28 21:06:17	raspberrypi2	3.18.11-v7+
#38	2015/09/28 21:01:10	raspberrypi2	3.18.11-v7+
#37	2015/09/28 21:00:42	raspberrypi2	3.18.11-v7+
#36	2015/09/28 21:00:02	raspberrypi2	3.18.11-v7+
#35	2015/09/28 20:55:30	raspberrypi2	3.18.11-v7+
#34	2015/09/28 20:27:03	raspberrypi2	3.18.11-v7+
#33	2015/09/28 20:22:43	raspberrypi2	3.18.11-v7+
#32	2015/09/28 20:21:07	raspberrypi2	unknown
#31	2015/09/28 20:18:34	raspberrypi2	unknown
#30	2015/09/28 20:08:36	raspberrypi2	unknown

# Run new LTP on Raspberry Pi2 (Cont'd)

- We can Show the log with Console output at run time



The screenshot shows the Jenkins web interface for a test run. On the left is a sidebar with navigation links: 'Back to Test / Test Suite', 'Status', 'Changes', 'Console Output' (highlighted with an orange callout bubble saying 'Select "Console Output"'), 'View as plain text', 'Edit Test Run Information', 'Delete Test Run', 'Parameters', 'Rebuild', and 'Previous Test Run'. The main area is titled 'Console Output' and displays a log of commands and their outputs. A green callout bubble labeled 'Test Log' points to this area. The log text is as follows:

```
Started by user anonymous  
Running remotely on raspberrypi2 in workspace /home/jenkins/buildzone  
[buildzone] $ /bin/sh -xe /tmp/hudson5564949394966131417.sh  
+ '[' '!' -d ../logs/Functional.LTP.all ']'  
+ echo testplan_default  
+ TESTPLAN=testplans/testplan_default.json  
+ source ../tests/Functional.LTP.all/ltp-all.sh  
++ source /home/jenkins/scripts/overlays.sh  
+++ . /home/jenkins/buildzone/./scripts/common.sh  
++++ assert_define JTA_ENGINE_PATH  
++++ varname=JTA_ENGINE_PATH  
++++ '[' -z /home/jenkins ']'  
++++ export JTA_ENGINE_PATH=/home/jenkins  
++++ JTA_ENGINE_PATH=/home/jenkins  
++++ export JTA_PARSER_PATH=/home/jenkins/scripts/parser  
++++ JTA_PARSER_PATH=/home/jenkins/scripts/parser  
++++ export JTA_LOGS_PATH=/home/jenkins/logs  
++++ JTA_LOGS_PATH=/home/jenkins/logs  
++++ TESTDIR=Functional.LTP.all  
++++ TEST_HOME=/home/jenkins/buildzone/./tests/Functional.LTP.all
```

# Run new LTP on Raspberry Pi2 (Cont'd)

## •To Show Test Results

Home Functional.LTP.20150420.all

Test History

Project Functional.LTP.20150420.all  
Linux Test Project.org Open POSIX Test Suite  
Disabled tests:  

- aio\_cancel/3-1
- aio\_cancel/7-1
- aio\_error/2-1
- aio\_fsync/5-1
- aio\_fsync/8-4
- shm\_open/23-1
- llo\_listio/2-1
- timer\_settime/5-2
- mq\_timedreceive/5-3
- mq\_timedsend/12-1

Test Results is SUCCESS !

Test Run History (trend)

- 2015/09/30 8:24:01 rasp2 / 3.18.11-v7+
- 2015/09/30 8:15:10 rasp2 / 3.18.11-v7+
- 2015/09/30 8:13:49 rasp2 / 3.18.11-v7+
- 2015/09/30 8:08:16 rasp2 / 3.18.11-v7+
- 2015/09/30 7:27:28 rasp2 / 3.18.11-v7+

Console output

```

++ echo FUNCTIONAL.LTP.2015
+ upName=FUNCTIONAL_LTP_20150420_ALL
+ fcname=FUNCTIONAL_LTP_20150420_ALL_FAIL_CASE_COUNT
+ fcc=
+ '[' -z '' ']'
+ return 0
POST BUILD TASK : SUCCESS
END OF POST BUILD TASK : 0
Finished: SUCCESS

```

Complete Running Test !



# Comparison result of running LTP

---

- For using OSS test suite effectively, we would like to find out the proper categorization to choose test case.
- We ran LTP for each cases and compared the results from the below perspectives.
  - Hardware difference: Minnow board vs Raspberry Pi2
  - Kernel difference: 3.18 vs 4.1
  - Userland difference: minimal vs with GUI
    - core-image-minimal vs core-image-sato (on Yocto Project)
  - Bit architecture difference: 32bit vs 64bit

# Result summary

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- **TPASS** - Indicates that the test case had the expected result and passed
- **TWARN** - Indicates that the test case experienced an unexpected or undesirable event that should not affect the test itself such as being unable to cleanup resources after the test finished.
- **TCONF** - Indicates that the test case was not written to run on the current hardware or software configuration such as machine type, or, kernel version.
- **TFAIL** - Indicates that the test case had an unexpected result and failed.
- **TBROK** - Indicates that the remaining test cases are broken and will not execute correctly, because some precondition not met, such as a resource not being available.

# Checking about TWARN/TFAIL

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- TWARN 3 items: Occurred on Minnow board only.
- TFAIL 3 items: The results of all cases are same. There might be no dependency.

# Checking about TBROK

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- The results of each cases are same, excepting the below.
  - 1 item: NOT occurred on Minnow board (32bit) only.
  - 1 item: Occurred on Raspberry Pi2 only.
  - 1 item: Occurred on core-image-minimal only.

# Checking about TCONF

case	1	2	3	4	5
Hardware	Minnow board (32bit)	Minnow board (64bit)	Raspberry Pi2	Raspberry Pi2	Raspberry Pi2
Kernel	4.1.8	4.1.8	3.18.11	4.1.10	4.1.10
Userland	core-image-sato	core-image-sato	core-image-sato	core-image-sato	core-image-minimal
TPASS	938	868	934	934	933
TWARN	3	3	0	0	0
TCONF	64	134	70	70	70
TFAIL	3	3	3	3	3
TBROK	54	54	55	55	56

- The results of each cases are same, excepting the below.
  - 10 items: NOT occurred on Minnow board (32bit) only.
  - 1 item: Occurred on Raspberry Pi2 only.
  - 2 items: Occurred on Minnow board only.
  - 66 items: Occurred on Minnow board (64bit) only.
  - 3 items: NOT occurred on Minnow board (64bit) only.

# The details of test case

- The below test case might be depending on Hardware. [\(7items\)](#)
  - Raspberry Pi2 only
    - clock\_getres01 (TCONF)
    - getrusage04 (TBROK)
  - Minnow board only
    - fanotify05, fanotify06 (TCONF)
    - Fanotify01, fanotify02, fanotify04 (TWARN)
- The below test case might be depending on bit architecture. [\(69items\)](#)
  - Minnow 64bit only
    - bdflush01, chown01\_16, chown02\_16, chown03\_16, chown05\_16, fchown01\_16, fchown02\_16, fchown03\_16, fchown05\_16, fstatat01, fstatat01\_64, getegid01\_16, getegid02\_16, geteuid01\_16, geteuid02\_16, getgid01\_16, getgid03\_16, getgroups01\_16, getgroups03\_16, getuid01\_16, getuid03\_16, lchown01\_16, lchown02\_16, modify\_ldt01, modify\_ldt02, modify\_ldt03, setfsgid01\_16, setfsgid02\_16, setfsgid03\_16, setfsuid01\_16, setfsuid02\_16, setfsuid03\_16, setfsuid04\_16, setgid01\_16, setgid02\_16, setgid03\_16, setgroups01\_16, setgroups02\_16, setgroups03\_16, setgroups04\_16, setregid01\_16, setregid03\_16, setregid04\_16, setresgid01\_16, setresgid02\_16, setresgid03\_16, setresgid04\_16, setresuid01\_16, setresuid02\_16, setresuid03\_16, setresuid04\_16, setresuid05\_16, setreuid01\_16, setreuid02\_16, setreuid03\_16, setreuid04\_16, setreuid05\_16, setreuid06\_16, setreuid07\_16, setuid01\_16, setuid02\_16, setuid03\_16, setuid04\_16 (TCONF)
  - Other than Minnow 64bit
    - fork14, getcpu01, mmap15 (TCONF)
- The below test case might be depending on User land. [\(1item\)](#)
  - core-image-minimal only.
    - Utimensat01 (TBROK)
- The below test case might be depending on Minnow 32bit. [\(11items\)](#)
  - Other than Minnow 32bit
    - eventfd01, io\_cancel01, io\_destroy01, io\_getevents01, io\_setup01, io\_submit01, readdir21, sgetmask01, set\_thread\_area01, ssetmask01 (TCONF)
    - syslog08 (TBROK)

# There is no items of depending on Kernel.

# Next step

---

- We sorted out the result of running test and chose test cases which might be depending on HW or Kernel or Userland or Bit architecture.
- As next step, we will check the below to review source code of them.
  - To confirm if assumption of dependency is correct.
  - To find out the regularity of description of source code.

# Conclusion

---

- Summary

- LTSI Test Framework has already had many kinds of target boards and Test suites.
- We showed How to Customize.
  - Adding a new target board such as Raspberry pi 2
  - Adding a new test suite such as LTP-20150420
- We showed the result of running LTP for each case.

- Future Works

- To add Kselftest to LTSI Test Project
- To think about making a SDK without yocto
- To think about how to choose proper test case.
  - Trying to check the result of running LTP



- LTSI project :
  - <http://ltsi.linuxfoundation.org/>
- LTSI Test project:
  - <http://ltsi.linuxfoundation.org/ltsi-test-project>
  - Test Framework:
    - <https://bitbucket.org/cogentembedded/jta-public.git>
- Linux Test Project
  - <http://ltp.sourceforge.net/>

